REMARKS

In the Official Action mailed on **13 September 2005** the Examiner reviewed claims 1-21. Claims 1-3, 8-10, and 15-17 were rejected under 35 U.S.C. §103(a) as being unpatentable over Haggar et al (USPub 2002/0091904, hereinafter "Haggar") in view of Shaylor (USPub 2002/0108025, hereinafter "Shaylor"). Claims 4-7, 11-14, and 18-21 were rejected under 35 U.S.C. §103(a) as being unpatentable over Haggar in view of Shaylor in view of Otis (USPub 2002/0099765, hereinafter "Otis").

Rejections under 35 U.S.C. §103(a)

Independent claims 1, 8, and 15 were rejected as being anticipated by Haggar in view of Shaylor.

Garbage collection, memory compaction and defragmentation typically require all processes or tasks to pause for a period of time. The simplest garbage collectors are non-incremental; they require the system to stop while garbage collection is performed. Incremental garbage collectors are able to interleave garbage collecting with activity from the rest of the system. Although the system can continue execution while the garbage collection is not yet complete, there still exists incremental pauses while the garbage collector executes. Some garbage collectors can theoretically run in parallel with the rest of the system, however, these garbage collectors implement barriers which block portions of memory while those portions of memory are in use by the garbage collector. Thus any process which requires the portion of memory being accessed by the garbage collector must be paused while the garbage collector finishes accessing the required memory (see http://en.wikipedia.org/wiki/Garbage_collection_%28computer_science%29, see Paul R. Wilson and Mark S. Johnstone, Real-Time Non-Copying Garbage Collection, page 1, see http://www.memorymanagement.org/glossary/, incremental garbage collection and parallel garbage collection).

Memory compaction is the process of shifting data that is in memory from one portion of memory to another portion of memory in order to create larger contiguous blocks of free memory. Memory compaction occurs as part of the defragmentation process (see http://en.wikipedia.org/wiki/Fragmentation). During the process of memory compaction the address of objects may change and therefore the value of the pointers may also change. For this reason, tasks must pause during memory compaction. Even if memory compaction occurs in stages, there must be some incremental pause by each task as the memory in use by the task is compacted.

Applicant respectfully points out that Haggar teaches a method whereby handles are stored in a stack at one end of the memory and storage blocks are stored in a stack at the opposite end of the memory. As more memory is allocated the two stacks grow towards each other (see Haggar, page 1, paragraph [0007] and see Haggar, FIG. 3). During memory deallocation and during garbage collection, memory is compacted (see Haggar, page 1, paragraph [0008], lines 9-17, see Haggar, page 4, paragraph [0039], lines 17-21, and see Haggar, page 5, paragraph [0041]-[0042]). Due to the nature of garbage collection and memory compaction, tasks that would normally be eligible to run must be paused for some period of time.

Applicant respectfully points out that Shaylor teaches moving tasks around in memory (see Shaylor, page 3, paragraph [0036], lines 4-6). Furthermore, Shaylor teaches memory compaction and defragmentation (see Shaylor, page 4, paragraph [0040], lines 1-2, and see Shaylor, page 4, paragraph [0042], lines 15-16). When more memory is required, Shaylor teaches moving a task's address space from one location in memory to another location in memory (see Shaylor, page 4, paragraph [0039] and see Shaylor, FIG. 5). Relocating objects in memory has a similar affect as memory compaction; pointers are updated and therefore tasks must be paused while the process is completed.

In contrast, the present invention divides the memory into independent sections (see paragraphs [0035]-[0036] and FIG. 3 of the instant application). Therefore, garbage collection, defragmentation and memory compaction all occur

without the need for any other tasks to be paused (see paragraph [0036], see paragraphs [0041]-[0042], and see paragraph [0052] of the instant application). Thus the present invention teaches a unique method which completely isolates one task space from another task space, thus allowing garbage collection of one task to occur without affecting any other task. The combination of Haggar and Shaylor teaches a method which interleaves tasks spaces and thus causes one task to halt another during memory maintenance processes such as memory compaction.

Accordingly, Applicant has amended independent claims 1, 8, and 15 to clarify that the present invention can perform garbage collection on an individual task's memory space without interfering with any other tasks. These amendments find support in FIGs. 2, and 3, paragraphs [0035-[0036], paragraphs [0041]-[0042], and paragraph [0052] of the instant application.

Hence, Applicant respectfully submits that independent claims 1, 8, and 15 as presently amended are in condition for allowance. Applicant also submits that claims 2-7, which depend upon claim 1, claims 9-14, which depend upon claim 8, and claims 16-21, which depend upon claim 15, are for the same reasons in condition for allowance and for reasons of the unique combinations recited in such claims.

CONCLUSION

It is submitted that the present application is presently in form for allowance. Such action is respectfully requested.

Respectfully submitted,

By 

Edward J. Grundler
Registration No. 47,615

Date: 19 October 2005

Edward J. Grundler
PARK, VAUGHAN & FLEMING LLP
2820 Fifth Street
Davis, CA 95616-7759
Tel: (530) 759-1663
FAX: (530) 759-1665
Email: edward@parklegal.com